Security Evaluation of a Control System Using Named Data Networking

Victor Perez, Mevlut Turker Garip, Silas Lam, and Lixia Zhang, *Fellow, IEEE*

***Abstract*—Security is an integral part of networked computer systems. The recent Named Data Networking (NDN) project aims to develop a new Internet architecture that communicates data using names rather than locations, the latter of which is what the current IP-based Internet does with IP addresses. One of the first real-world applications using NDN is a lighting control system. We conduct a red team assessment of the current state of the security of this lighting system and its NDN implementation. The system is representative of a more general class of automated controller systems. Our analysis found that due to NDN's use of named data, the system inherently prevents most attacks that IP-based systems are vulnerable to. Although many parts of the system are secure, we discovered some problems with the verification of timestamps and processing of large packets that led to a severe memory leak. The system also lacks a secure key distribution mechanism. While NDN security is on the right track, there are important security design issues NDN must account for.**

***Index Terms*—Computer networks, Computer security, Building automation**

I. Introduction

MODERN buildings rely on state of the art Building Automation and Control Systems (BACS). They provide centralized control over core building services that affect the overall building environment. Some of these essential services include heating, ventilation, air conditioning (HVAC), and lighting. Before BACS, human operators would survey building conditions manually. This would serve as input to decide how to calibrate air and lighting systems in the building. However, this is a tedious and imperfect way to regulate building conditions.

BACS can collect data from building sensors to help regulate a building's environment as shown in Figure 1; thus, improving energy efficiency. As a result, a key element of these systems is communication. The controller in BACS can be either a human and/or machine process. They must be able to communicate with sensors to get building data. Once they assess the current conditions, they need to communicate with remote controllers or actuators capable of physically altering building conditions, such as an air conditioning unit.

To facilitate this communication, many BACS today make use of the BACnet or LonWorks protocols, which allow for communication of sensors, controllers, and actuators. [1] explains that increasingly the Internet Protocol (IP) is used as the backbone of modern BACS within individual systems and also to integrate systems such as lighting and HVAC. As discussed in [2] BACS originated in a time when security was not a primary concern. This was fine when BACS were operating as *closed* systems, but now that IP is used to integrate them and expose them to external control interfaces on the Internet, the threat model should be reassessed. One solution to secure BACS that use IP is to segregate BACS using VLANs. This unfortunately is not an ideal solution, since it segregates the systems that are trying to self-calibrate and exchange sensor data.

The authors of [1] discuss an alternative to running BACS over IP. They present their design and implementation of an authenticated lighting control system that uses Named Data Networking (NDN) for communication. NDN uses a data-centric model, unlike IP's host-centric model. One of the benefits of using NDN comes when addressing specific BACS components. Instead of using an arbitrarily assigned IP address, meaningful names can be used by the control application to address a device. Another potential benefit is that NDN may be better suited to address the problem of security in BACS, since the authors used authenticated commands to control the lighting system.

We conduct a red team[1] evaluation on a prototype of the lighting system described in [1] to assess the security of the system. The goal of this effort is a first step towards understanding the security strengths and the remaining vulnerabilities of the NDN architecture, as well as its early prototype implementations. In this paper we present our security experiments and evaluation findings. The rest of the paper is organized as follows. Section II is a brief explanation of NDN packets, and section III presents the hardware and software setup of the NDN lighting control system. In Section IV we discuss test environment limitations. Sections V though VIII describe our security experiments and present the corresponding results. We finish with a discussion on future work in Section IX and conclusion in Section X.

II. NDN Communication

Communication in Named Data Networking (NDN) consists of the exchange of *Interest* packets and *Data* packets as described in [3]. To receive data a consumer must express interest by sending out an Interest packet that carries a name identifying the piece of data they want. NDN routers will then

V. Perez, M. Garip, S. Lam, and L. Zhang are with the Department of Computer Science, University of California Los Angeles, Los Angeles, CA, 90095 USA e-mail: vperez@cs.ucla.edu; mtgarip@cs.ucla.edu; silaslam@cs.ucla.edu; lixia@cs.ucla.edu.

[1]Team of testers that assess the security of an organization or system, which is often unaware of the existence of the team or the exact assignment.

978-1-4799-1270-4/13$31.00

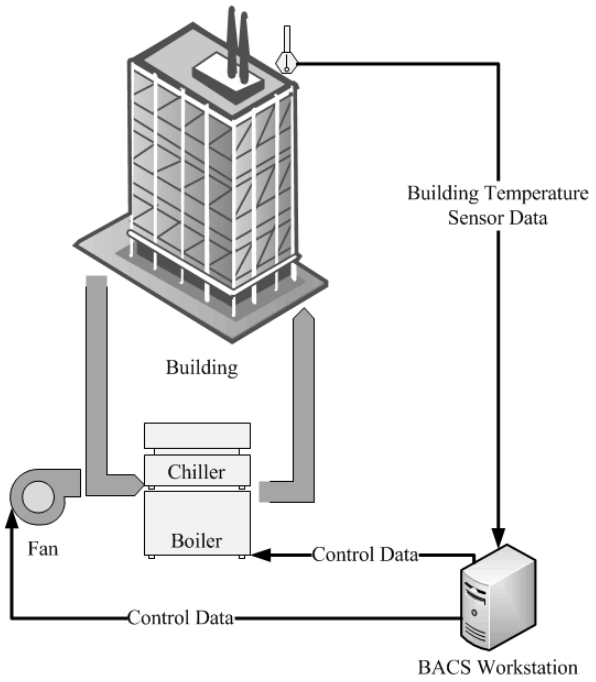

Fig. 1. Sample Building Automation and Control System.

forward Interest packets to the appropriate data producers. When a data producer receives an Interest packet a Data packet is generated for the request and it is sent to the consumer. The structure of a typical NDN Interest packet consists of the content name, a selector and nonce value. An NDN Data packet consists of a content name and data payload. In addition, because Data packets are signed they also contain a signature and signature data. Signing of Data packets allows the data consumers to verify they are receiving the data requested.

In the lighting control system we evaluate, Interest packets are also signed. This is so that the controller board can authenticate that they came from an authorized operator machine.

III. Lighting Control System Testbed

Here we describe the hardware and software setup of the lighting control system. The system prototype we were given is slightly different from what is described in [1].

A. Hardware

The system hardware consists of three primary components: the lighting fixture, the operator machine, and the controller board. The lighting fixture is an array of red, blue and green LED light bulbs, which can be digitally powered on or off via a serial interface. This serial interface connects to the controller machine, which is an Overo Gumstix board with an ARMv7a processor and 512MB RAM, running the Angstrom 2011.03 Linux distribution. The controller board also connects to the operator machine, which can initiate changes to the lighting fixture via NDN interest packets. The operator machine is a standard PC with an Intel Core i5 processor and 8 GB of RAM. It runs the Ubuntu 12.04 32-bit Linux distribution.

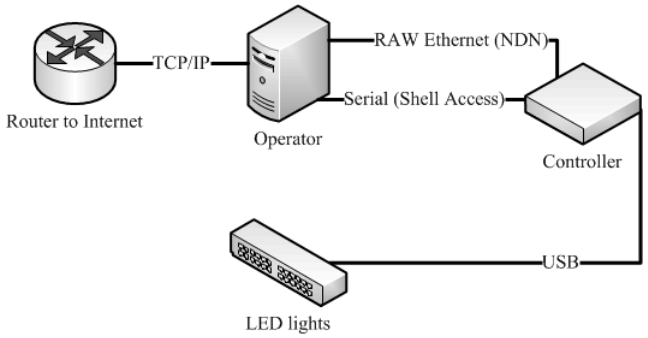

Fig. 2. Lighting Control Testbed

The operator machine connects to the controller via a USB interface, which allows console access to the controller. In addition, the controller and operator machines are connected using a RAW ethernet interface. This network interface is used to exchange NDN packets that carry light control commands. The operator machine also has an additional network interface that connects to the Internet to allow us to connect remotely. Figure 2 shows the testbed.

B. Software

To communicate and exchange lighting commands using NDN, CCNx[2] was compiled and installed on both the operator and controller machines. Typically, NDN packets generated by CCNx are tunneled over IP between machines. However, in the lighting control testbed a link layer protocol called NDNLP[3] is used to transmit NDN packets. NDNLP can run as a user-space daemon and can be used to deliver NDN packets over Ethernet links and TCP/UDP tunnels. In this case it is configured for Ethernet links. According to the authors of NDNLP, it is beneficial to run directly over Ethernet without IP as this eliminates the extra layer of processing [4]. NDNLP provides support for sending link level acknowledgements, and we experimented with this feature in our tests.

The actual lighting control application was written in Python. The application is split into two modules. One module, *sequencer.py*, runs on the operator machine, and it is used to send NDN interest packets to the controller. The second module, *interface.py*, runs on the controller board. It registers to listen for a specific NDN prefix and waits for arriving interests that match.

In this system implementation, NDN interests are signed by the operator with a private key. It is important to note that signed interests are not part of the original NDN design, but that this feature was added by the authors of [1]. Upon the arrival of a matching NDN packet, the controller module will then authenticate the packet with the operator's public key. If the packet is successfully verified, the operator will then execute the lighting command encoded in the NDN interest packet. Listing 1 shows the XML syntax of a sample NDN interest command packet. The first five components in the name make up the NDN application prefix. The next

[2]https://github.com/ProjectCCNx/ccnx
[3]https://github.com/NDN-Routing/NDNLP

component is the light name. The seventh and eighth name components specify the application lighting function to be executed at the controller board. This is followed by the next component, which is a function parameter specifying the light colors which should be turned on or off by the light fixture. The last two name components encapsulate the operator's public key, and the signature bits obtained by signing the previous name components using the operator's private key. After the name element a nonce value is appended at the end of the interest packet.

```
<Interest>
  <Name>
    <Component>ndn</Component>
    <Component>ucla.edu</Component>
    <Component>lighting</Component>
    <Component>redteam</Component>
    <Component>example</Component>
    <Component>bh4805</Component>
    <Component>rgb-8bit-hex</Component>
    <Component>setRGB</Component>
    <Component>ff0000</Component>
    <Component>Public key,timestamp</Component>
    <Component>Signature Bits</Component>
  </Name>
  <Nonce>baec9c26e400a81d699fab36</Nonce>
</Interest>
```

Listing 1. Sample lighting command format

IV. Testing Environment Limitations

Although we were able to conduct several experiments on the lighting controller test environment, there were some limitations that prevented us from fully testing our security attacks. We briefly mention some of these limitations.

Our team was given direct access to the operator machine. However, as shown in Figure 2, the operator and controller board communicate with each other on a closed raw Ethernet network. This limits our testing efforts because all of our attacks have to initiate at the operator machine. In a more realistic scenario we envision that the operator and controller network might be formed by a series of connected routers and/or switches. This allows for attacks that start at external devices on the network. In addition, the use of routers between the operator and controller could present other potential security vulnerabilities, which we were not able to explore given our testbed.

To mitigate some of these limitations, our approach was to simulate the behavior we would expect if we had a more complete and realistic test environment. For example, for an attack in which packets destined for the controller are intercepted by an attacker on the network, we disable the NDNLP daemon. This simulates packets being intercepted, even though the direct Ethernet link between the operator and controller machines would prevent such an attack in the current testbed. In the following sections we describe our security experiments and present the results.

V. Replay Command Attack

In this attack we intercept NDN interest packets as they are being sent from the operator machine to the controller board. We then use the stored packet and attempt to force the controller board to execute the command encoded in the packet. This is a typical example of a man-in-the-middle attack (MITM), in which an attacker eavesdrops on the network conversation between two victims, which in our case are the operator machine and controller board. However, given the current lighting controller system, it is not possible to initiate independent connections with the victim hosts. Instead, as will be explained in the next subsection, we simulated this MITM attack.

It is important to note that the current lighting controller application code does have countermeasures to defend itself against replay attacks like this. As shown in Listing 1, the interest component before the signature, a timestamp value is inserted by the operator to let the controller know when the command was issued. This way if a packet is replayed after a long delay, the controller board can choose to ignore it. A configuration file is used by the controller lighting module to specify the time window tolerance, δ. Given a packet arrival time at the controller, τ_c, and a packet send timestamp inserted by the operator, τ_o, the lighting controller will reject the command if the following equation does not hold.

$$|\tau_c - \tau_o| \leq \delta \tag{1}$$

A. Experiment

On the operator machine the CCNx and NDNLP daemons are started. We then set up tcpdump[4] to listen on the outgoing NDN interface. The tool is started with the appropriate filters to capture only outgoing NDN packets. Because we cannot establish independent connections to the operator and controller, we simulate intercepting packets by stopping the CCNx and NDNLP processes on the controller board.

Next, the lighting control module on the operator is invoked to send a command to the controller and change the color of the lighting fixture to red. Tcpdump will capture the packet to a file, and although the packet will be received by the controller board, the NDNLP process will not receive it.

With the packet captured, the NDNLP and CCNx daemons at the controller are started along with the controller application module. The δ tolerance time had been configured to five minutes. We then use tcpreplay[5] on the operator machine to read in the captured packet and inject it into the NDN network interface.

B. Discussion

When the replayed packet arrived at the controller, the application module verified the packet signature and then checked that equation (1) was true. After verifying, the controller accepted the packet and executed the specified function to change the lighting fixture color to red.

Although the application code running on the controller does protect itself against replay attacks, we were still successful executing this attack because the value of δ was configured sufficiently large, and because the operator and controller clocks were not synchronized, due to clock drift error.

[4]http://www.tcpdump.org/

[5]http://tcpreplay.synfin.net/

One approach to prevent this replay attack is to minimize the value of δ. Unfortunately, this alone cannot solve the problem because clock drift will result in the operator and controller clock being out of sync. This could be enough to fool the controller application into executing a replayed packet.

A more promising solution could be to make use of the NTP[6] time synchronization protocol which would address the problem of clock drift between the two machines. Yet, even this is not a perfect solution. As the author of [5] shows, NTP is also vulnerable to security attacks. The root cause of this attack is that the lighting control application is relying exclusively on external sources to authenticate valid commands. This problem however, can be mitigated by using timestamps together with sequence numbers. Sequence numbers can be negotiated by the controller and operator, and they would serve to uniquely identify both the arriving command packets at the controller and the acknowledgement data packets sent to the operator. Together with timestamp information a replay attack packet can be detected by the controller. The findings of this attack are applicable to any NDN BACS that only uses timestamps to mitigate MITM attacks.

VI. Link Layer Acknowledgement Attack

In our testbed, NDN runs over NDNLP instead of TCP/UDP/IP tunnels. This is advantageous because it eliminates the task of configuring IP addresses and the extra processing associated with it. NDNLP addresses two major issues: (1) messages longer than the Ethernet MTU cannot be sent, and (2) dealing with packet loss. The first issue is solved with fragmentation and reassembly, while the second issue is dealt with by using acknowledgements and retransmissions. It is the latter that we try to exploit in this attack.

NDNLP uses acknowledgements and retransmissions in order to handle transmission errors on the physical link. Each link data packet has its own associated sequence number. It uses a cumulative acknowledgement scheme, which means that a single acknowledgement packet can acknowledge multiple link data packets. This is done by using a bitmap in the acknowledgement packet. If a packet is not acknowledged in time, the sender can retransmit. The NDNLP daemon can be configured to turn off/on acknowledgements, as well as settings for retry counts and retransmission times.

```
<NdnlpData>
  <NdnlpSequence>sequence number</NdnlpSequence>
  <NdnlpFlags>flags</NdnlpFlags>
  <NdnlpFragIndex>fragment index</NdnlpFragIndex>
  <NdnlpFragCount>fragment count</NdnlpFragCount>
  <NdnlpPayload>payload</NdnlpPayload>
</NdnlpData>
```

Listing 2. Sample NDNLP Link Acknowledgement Packet

Listing 2 shows the XML structure of a link data packet. Note that only a sequence number identifies a link data packet, ignoring fragmentation. There is no timestamp associated with a link data packet. It is left to the upper layers to handle the timeout of old packets. Retransmitted packets are exactly the same as the original packet, including payload and sequence number.

[6]http://tools.ietf.org/html/rfc958

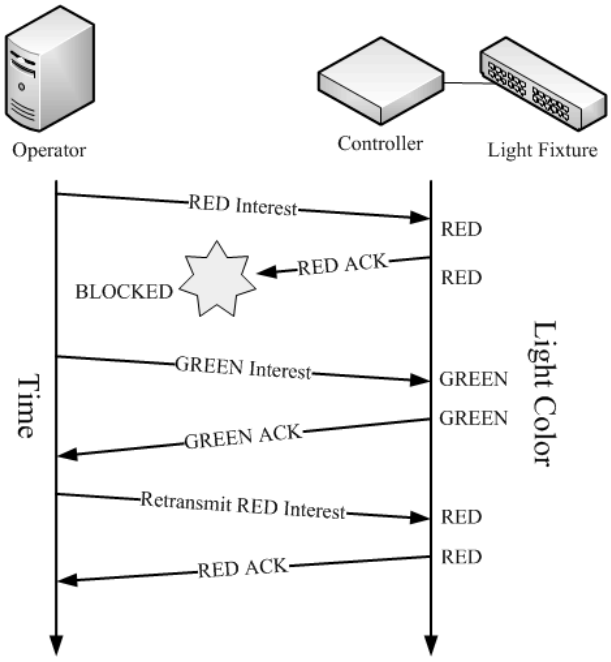

Fig. 3. NDNLP Ack Attack

In traditional NDN, interests are sent from consumer host A to a producer host B. Host B processes the interest and replies back with a data packet. In the NDN lighting BACS, the producer host B also has a side effect upon receiving a valid interest, such as turning a light color to red. These side effects are time sensitive, as a single light can become multiple colors within a second. Two cases for this would be in a party/disco scene with strobe lights, and a scene that requires a slowly fading light. NDN itself does not guarantee in-order packet delivery, and the NDN lighting BACS also does not process interests in-order. In our experiment, we wanted to exploit the NDNLP acknowledgements/retransmissions, keeping in mind that the lighting side effects are time sensitive.

In the following attacks, our assumption is that the attacker can block packets, be it an interest packet or the link data acknowledgement packet. This is possible in a man-in-the-middle scenario. It is important to note that we turned off cumulative acknowledgements so that an acknowledgement is associated with a single link data packet. This was done to simplify the experiment procedure.

A. Experiment

Figure 3 shows the proposed attack scenario. The operator wishes to change the lights from red to green, one after another. The red ack gets blocked, so the operator retransmits the red interest after the green interest is sent. The end result is that the light ends with red instead of the expected green.

B. Discussion

We conducted our above scenario, but did not obtain the anticipated result. The controller received the retransmitted red interest and sent the acknowledgement, but it did not change the light color, leaving it green. The NDNLP daemon keeps

track of link data packets it has received. When the daemon receives a packet it has already seen, it responds with an ack, but does not forward the packet to the upper application layer since it knows it is a duplicate packet. We have not explored how many packets it tracks this way. This exploit attempt was unsuccessful.

In another scenario, we could block the original red interest instead of its acknowledgement. In this case, the end result would be a red light. However, this is equivalent to the packet reorder attack presented in Section VIII.

VII. Memory Leak Exploit Discovery

One of our goals was to test the limits of the lighting control system by targeting both the application level code and the network (CCNx) code. This should expose oversight in corner case testing that might otherwise go unnoticed until an attacker decides to probe for such a vulnerability. Specifically, in this attack we increase the size of an NDN interest packet and observe how the system handles large size packets.

```
<Interest>
  <Name>
    <Component>ndn</Component>
    <Component>ucla.edu</Component>
    <Component>lighting</Component>
    <Component>redteam</Component>
    <Component>example</Component>
    <Component>bh4805</Component>
    <Component>rgb-8bit-hex</Component>
    <Component>setRGB</Component>
    <Component>ff0000</Component>
    <Component>Random Component 1</Component>
    <Component>Random Component 2</Component>
                .
                .
                .
    <Component>Random Component 4000</Component>
    <Component>Public key,timestamp</Component>
    <Component>Signature Bits</Component>
  </Name>
  <Nonce>baec9c26e400a81d699fab36</Nonce>
</Interest>
```

Listing 3. Large NDN interest command packet

A. Experiment

Typical NDN packets exchanged between the operator and controller board are relatively small at 509 bytes, including the NDNLP and Ethernet headers. In the first phase of this experiment we wrote an attack script on the operator machine that would send 2,000 properly formatted command packets requesting a light color change. We execute the script on the operator machine and use the Linux top command to probe the system statistics on the operator board.

We then test how the system would handle fewer, but much larger NDN interest packets. For this a module was written to generate properly formatted NDN packets with a much larger payload. One obvious way to increase the packet size is to increase the length of the NDN Interest name. However, generating arbitrary NDN packets with long names would not work since the CCNx daemon would discard packets for which it cannot respond to. We wrote a program that would generate proper lighting controller command packets as shown in Listing 1, and then injected an arbitrary number of name components before the packet component that contains public key and timestamp information. The data contained in these additional name components is not important; what matters is that the newly generated packets are relatively large. Random nonce values are appended as the extra name components. Listing 3 shows a sample of this large attack packet. In our experiment, we chose to append 4,000 name components before the components containing the public key and timestamp information. Because of the size of the packets, NDNLP will fragment the packet into smaller packets before sending it over Ethernet. The packet is fragmented into 31 smaller packets of approximately 1,514 bytes, which are reassembled by the NDNLP daemon at the controller board before being delivered to the CCNx process.

The packet generator module is invoked from the operator machine to send this large packet to the controller. Just as in the first part of this experiment, we start the Linux top command to probe for any system anomalies in the controller board.

B. Discussion

As explained in the previous section, the first part of this experiment bombards the controller board with 2,000 properly formatted NDN interests packets requesting a change to the lighting fixture color. The test takes several minutes to execute. During the experiment we monitored the controller board. No abnormal behavior was observed during the duration of the test. Thus, we conclude that the lighting system is robust against a steady stream of incoming operator commands. This was expected, as the lighting control is required to be able to handle many light change requests in small time intervals.

We discovered a critical exploit when conducting the second phase of the experiment. The flaw is not a problem with the controller application module, but with the CCNx process running on the controller. After sending 5 sequential large packets shown in Listing 3, the CCNx daemon was killed by the operating system, and the application was terminated.

Repeating the experiment we concluded that the process at fault was *ccnd*, the CCNx daemon. After analyzing the operating system's kernel logs, we traced the problem to *ccnd* being terminated because it consumes all of the system's memory resources. Using the top command we repeated the experiment and carefully profiled how each large packet sent by the operator significantly increased the memory usage by *ccnd*. It is now clear that we have discovered a memory leak in CCNx. This is a serious flaw because the same attack can be launched against any *ccnd* daemon process listening for incoming interests. The attack shows that this early stage implementation of CCNx has not been as extensively debugged and tested as the current Internet stack. Thus we suspect there are other such implementation vulnerabilities in CCNx.

VIII. Packet Reorder Attack

Unlike TCP, NDN does not guarantee in-order packet delivery. This is not necessarily a disadvantage, as IP's success is due to its simplicity and weak demands to the lower layers,

namely: stateless, unreliable, unordered, best-effort delivery. NDN was developed with these features and IP's success in mind [3].

If an NDN application requires in-order packet delivery, it is the responsibility of the application to implement this functionality. Unfortunately, the NDN lighting BACS does not enforce in-order packet delivery, possibly because it was not considered necessary in the original design. This leads us to exploit a vulnerability and potentially change the lights to an unintended color.

In this attack scenario, we consider the case where we have a MITM that can delay arbitrary packets with the end result of packets arriving out of order. We begin by assuming we want the lights to remain a certain color for a period of time. When a legitimate operator begins sending a sequence of interest packets with light commands, we wait until we see a packet that contains the color we want. We then delay this packet such that it is at the end of the sequence. As a result, the lights stay at the color we want. This attack works as long as the delayed packet is within the time window tolerance as explained in Section V.

A countermeasure to this attack is to enforce in-order delivery at the application level. This could be done by using sequence numbers in the interests and reordering the interests upon reception. This additional step of processing may be acceptable in more correct-sensitive applications. However, the processing delay could negatively affect time-sensitive applications. Guaranteeing in-order delivery at a lower level would be more effective, but not all applications have a need for such a feature.

IX. Future Work

A. Key Exchange Considerations

The NDN architecture depends on the usage of cryptography for security. In its current implementation, the challenge of secure key distribution has not been completely solved. Because the current NDN architecture has not addressed this issue, the lighting system hardcodes the operator's public key in the controller to validate arriving interests. The interest packet itself also contains a public key, but this is only used to match with the list of accepted keys on the controller.

The above mentioned design works properly if the operator always uses the same public key, but it does not provide any secure procedure to update the keys without manual access to both the operator and the controller. Good cryptography practices state that key pairs should be updated after a period of time because an attacker is more easily able to predict a key the longer it is in use. The NDN lighting BACS currently ignores this problem. However, we expect key update functionality to be implemented in the future. We do not believe that key distribution via certificate authorities is a good solution because it will not scale well if there are many NDN users. A future decentralized key distribution mechanism would be preferred. Whatever the solution, key distribution must be safeguarded against potential MITM attacks during the key exchange process. It is especially important that NDN address this problem because the architecture makes extensive use of keys to sign packets.

B. Revisit CCNx Memory Flaw

We hope to present our findings to the CCNx development team. After they fix the memory leak, an even more elaborate large packet generation attack can be devised to verify that the problem was properly corrected. It would be useful to use memory profiling tools like valgrind to conduct a detailed analysis on the memory footprint of the CCNx and NDNLP processes. Given that both projects are still under heavy development, it is possible that memory bugs still exist.

X. Conclusion

Building Automation and Control Systems (BACS) continue to be deployed to regulate the environment of buildings. Of great concern is the security of these systems. The very first BACS to run on Named Data Network (NDN) is a lighting controller. We were interested in the potential security improvements and security design lessons.

In our research as the red team for the lighting control system, we devised a replay command attack. Our results suggest that although the application software was written with replay attack countermeasures, the fact that it relies on timestamps keeps it vulnerable. This external source dependence should be minimized. We also conducted an attack against the link layer protocol, NDNLP, to explore whether we could fool the operator in resending unacknowledged packets. It turns out that NDNLP will detect duplicate packets, and it will ignore them. This attack was unsuccessful. We also described a packet reorder attack, in which an attacker can manipulate the order of the command packets resulting in an unwanted effect at the lighting controller.

Another important contribution of our tests was the discovery of a memory leak flaw in the CCNx code. By using large packets we exposed a problem that may have otherwise gone unnoticed. In addition, we also concluded that NDN does not have a secure key distribution mechanism. Because the architecture depends heavily on keys, a solution needs to be proposed soon.

There is still further work that can be done to gain a better understanding of the state of security of the lighting controller and NDN. Running BACS over NDN has clear advantages, but there are some security issues that should be addressed.

Acknowledgment

The authors would like to thank Alex Horn and Wentao Shang for setting up and supporting the testbed. We also thank Alex Afanasyev for adding NDNLP support to ndndump.

References

[1] J. Burke, A. Horn, and A. Marianantoni, "Authenticated lighting control using named data networking," University of California, Los Angeles, Los Angeles, CA, NDN Technical Report NDN-0011, Oct. 2012.

[2] W. Granzer, F. Praus, and W. Kastner, "Security in building automation systems," *IEEE Trans. Ind. Electron.*, vol. 57, pp. 3622–3630, Nov. 2010.

[3] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proc. CoNEXT '09*, Rome, Italy, Dec. 1–4, 2009.

[4] J. Shi and B. Zhang, "Ndnlp: A link protocol for ndn," The University of Arizona, Tucson, AZ, NDN Technical Report NDN-0006, Jul. 2012.

[5] M. Bishop, "A security analysis of the ntp protocol version 2," *Computer Security Applications Conference*, pp. 20–29, Dec. 1990.